Abstract Syntax Networks for Code Generation and Semantic Parsing

Berkeley

Maxim Rabinovich*, Mitchell Stern*, Dan Klein
Code Generation

[Image: A screenshot from the game of Hearthstone, showing a turn in progress with various cards and units.]

[Ling et al. 2016]
class DireWolfAlpha(MinionCard):
    def __init__(self):
        super().__init__("Dire Wolf Alpha", 2, CHARACTER_CLASS.ALL, CARD_RARITY.COMMON, minion_type=MINION_TYPE.BEAST)

    def create_minion(self, player):
        return Minion(2, 2, auras=[
            Aura(ChangeAttack(1), MinionSelector(Adjacent()))
        ])
class DireWolfAlpha(MinionCard):
    def __init__(self):
        super().__init__(
            "Dire Wolf Alpha", 2,
            CHARACTER_CLASS.ALL,
            CARD_RARITY.COMMON,
            minion_type=MINION_TYPE.BEAST)

    def create_minion(self, player):
        return Minion(2, 2, auras=[
            Aura(ChangeAttack(1),
            MinionSelector(Adjacent()))
        ])
class DireWolfAlpha(MinionCard):
    def __init__(self):
        super().__init__(
            "Dire Wolf Alpha", 2,
            CHARACTER_CLASS.ALL,
            CARD_RARITY.COMMON,
            minion_type=MINION_TYPE.BEAST)

    def create_minion(self, player):
        return Minion(2, 2, auras=[
            Aura(ChangeAttack(1),
                MinionSelector(Adjacent()))
        ])
class DireWolfAlpha(MinionCard):
    def __init__(self):
        super().__init__(
            "Dire Wolf Alpha", 2,
            CHARACTER_CLASS.ALL,
            CARD_RARITY.COMMON,
            minion_type=MINION_TYPE.BEAST)

    def create_minion(self, player):
        return Minion(2, 2, auras=[
            Aura(ChangeAttack(1),
                 MinionSelector(Adjacent()))
        ])
class DireWolfAlpha(MinionCard):
    def __init__(self):
        super().__init__(
            "Dire Wolf Alpha", 2,
            CHARACTER_CLASS.ALL,
            CARD_RARITY.COMMON,
            minion_type=MINION_TYPE.BEAST)

    def create_minion(self, player):
        return Minion(2, 2, auras=[
            Aura(ChangeAttack(1),
            MinionSelector(Adjacent()))
        ])
The Main Idea
Card Representation

<table>
<thead>
<tr>
<th>attribute</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Dire Wolf Alpha</td>
</tr>
<tr>
<td>cost</td>
<td>2</td>
</tr>
<tr>
<td>type</td>
<td>Minion</td>
</tr>
<tr>
<td>rarity</td>
<td>Common</td>
</tr>
<tr>
<td>race</td>
<td>Beast</td>
</tr>
<tr>
<td>class</td>
<td>Neutral</td>
</tr>
<tr>
<td>health</td>
<td>2</td>
</tr>
<tr>
<td>attack</td>
<td>2</td>
</tr>
<tr>
<td>description</td>
<td>Adjacent minions have +1 Attack.</td>
</tr>
</tbody>
</table>

class DireWolfAlpha(MinionCard):
    def __init__(self):
        super().__init__(
            "Dire Wolf Alpha", 2,
            CHARACTER_CLASS.ALL,
            CARD_RARITY.COMMON,
            minion_type=MINION_TYPE.BEAST)

    def create_minion(self, player):
        return Minion(2, 2, auras=[
            Aura(ChangeAttack(1),
            MinionSelector(Adjacent()))
        ])
Aura(ChangeAttack(1), MinionSelector(Adjacent()))
Code Representations

Aura(ChangeAttack(1), MinionSelector(Adjacent()))

Aura ( ChangeAttack ( 1 ) , MinionSelector ( Adjacent ( ) ) )
Aura(ChangeAttack(1), MinionSelector(Adjacent()))
Model Overview

Input

Encoding

Output

Adjacent minions have +1 Attack.

Dire Wolf Alpha

Beast
Encoding
<table>
<thead>
<tr>
<th><strong>name</strong></th>
<th>Dire Wolf Alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cost</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>Minion</td>
</tr>
<tr>
<td><strong>rarity</strong></td>
<td>Common</td>
</tr>
<tr>
<td><strong>race</strong></td>
<td>Beast</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td>Neutral</td>
</tr>
<tr>
<td><strong>health</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>attack</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>description</strong></td>
<td>Adjacent minions have +1 Attack.</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td>Dire Wolf Alpha</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>cost</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>Minion</td>
</tr>
<tr>
<td><strong>rarity</strong></td>
<td>Common</td>
</tr>
<tr>
<td><strong>race</strong></td>
<td>Beast</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td>Neutral</td>
</tr>
<tr>
<td><strong>health</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>attack</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>description</strong></td>
<td>Adjacent minions have +1 Attack.</td>
</tr>
<tr>
<td>name</td>
<td>Dire Wolf Alpha</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>cost</td>
<td>2</td>
</tr>
<tr>
<td>type</td>
<td>Minion</td>
</tr>
<tr>
<td>rarity</td>
<td>Common</td>
</tr>
<tr>
<td>race</td>
<td>Beast</td>
</tr>
<tr>
<td>class</td>
<td>Neutral</td>
</tr>
<tr>
<td>health</td>
<td>2</td>
</tr>
<tr>
<td>attack</td>
<td>2</td>
</tr>
<tr>
<td>description</td>
<td>Adjacent minions have +1 Attack.</td>
</tr>
<tr>
<td>name</td>
<td>Dire Wolf Alpha</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>cost</td>
<td>2</td>
</tr>
<tr>
<td>type</td>
<td>Minion</td>
</tr>
<tr>
<td>rarity</td>
<td>Common</td>
</tr>
<tr>
<td>race</td>
<td>Beast</td>
</tr>
<tr>
<td>class</td>
<td>Neutral</td>
</tr>
<tr>
<td>health</td>
<td>2</td>
</tr>
<tr>
<td>attack</td>
<td>2</td>
</tr>
<tr>
<td>description</td>
<td>Adjacent minions have +1 Attack.</td>
</tr>
</tbody>
</table>

![Diagram](image-url)
<table>
<thead>
<tr>
<th>name</th>
<th>Dire Wolf Alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost</td>
<td>2</td>
</tr>
<tr>
<td>type</td>
<td>Minion</td>
</tr>
<tr>
<td>rarity</td>
<td>Common</td>
</tr>
<tr>
<td>race</td>
<td>Beast</td>
</tr>
<tr>
<td>class</td>
<td>Neutral</td>
</tr>
<tr>
<td>health</td>
<td>2</td>
</tr>
<tr>
<td>attack</td>
<td>2</td>
</tr>
<tr>
<td>description</td>
<td>Adjacent minions have +1 Attack.</td>
</tr>
<tr>
<td>name</td>
<td>Dire Wolf Alpha</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>cost</td>
<td>2</td>
</tr>
<tr>
<td>type</td>
<td>Minion</td>
</tr>
<tr>
<td>rarity</td>
<td>Common</td>
</tr>
<tr>
<td>race</td>
<td>Beast</td>
</tr>
<tr>
<td>class</td>
<td>Neutral</td>
</tr>
<tr>
<td>health</td>
<td>2</td>
</tr>
<tr>
<td>attack</td>
<td>2</td>
</tr>
<tr>
<td>description</td>
<td>Adjacent minions have +1 Attack.</td>
</tr>
</tbody>
</table>
Decoder Architecture
Decoder Architecture
Decoder Architecture
Decoder Architecture

expr

BinOp  Call  Name  ...

Adjacent minions have +1 Attack.
Decoder Architecture

expr

?  
BinOp  Call  Name  ...

Call
Decoder Architecture
Decoder Architecture

expr
BinOp Call Name ...

Call
expr
expr*

Call
func args
expr expr*

identifier
"x" "Aura" ...
Decoder Architecture

expr

BinOp  Call  Name ...

Call

Call

func

args

expr

expr*

identifier

"x" "Aura" ...

"Aura"
Decoding Process
Decoding Process
Decoding Process
Decoding Process

Call

expr

func

args

expr*

Call

func

args

Dire Wolf Alpha

Adjacent missions have +1 Attack.
Decoding Process

Call

func

args

expr

expr*

Call

func

args

Dire Wolf Alpha

Adjacent melees have +1 Attack.
Decoding Process
Decoding Process
Decoding Process

expr

Call

func

args

expr*

expr

BinOp  Call  Name  ...

Dire Wolf Alpha

Adjacent minions have +1 Attack.
Decoding Process
Decoding Process

Call
- func
- args
  - expr*

Name
- id

identifier

expr
- BinOp
- Call
- Name

...
Decoding Process
Decoding Process

```
Decoding Process

Call
  func
  args
    id
    "Aura"
    expr*

```

Adjacent missions have +1 Attack.
Decoding Process

Diagram showing a call structure with a `Name` node labeled "Aura", a `func` node, and an `args` node with an `expr*` node. There is also a decision node labeled `expr*` with options for `Next` and `Stop`. A card titled "Dire Wolf Alpha" is present, indicating "Adjacent minions have +1 Attack."
Decoding Process

Call

- Name
  - id
  - "Aura"
- func
- args
  - expr*

expr*

Diamond

Next
Stop

Card:
- Name: Dire Wolf Alpha
- Text: Adjacent minions have +1 Attack
- Cost: 2
Decoding Process

```
Expr
  Call
    func
      Name
        id
        "Aura"
    args
  expr
  expr*
```

Card:

```
Dire Wolf Alpha
Adjacent minions have +1 Attack.
```
Decoding Process
Decoding Process

![Decoding Process Diagram]

1. **Call**
   - **func**
   - **args**
   - **Name**
     - **id**
     - "Aura"
   - **Call**
     - **func**
     - **args**
     - **Name**
       - **id**
       - "ChangeAttack"
     - **Num**
       - **n**
       - **1**

2. **expr**
   - **?**
   - **Next**
   - **Stop**

3. **Card**
   - **Dire Wolf Alpha**
   - **Adjacent missions have +1 Attack**
   - **Cost:** 2
Decoding Process

[Diagram of a tree structure with nodes labeled as 'Call', 'func', 'args', 'Name', 'id', '"Aura"', 'Call', 'func', 'args', 'Name', 'id', '"ChangeAttack"', 'Num', 'n', '1', 'expr*', '?', 'Next', 'Stop', 'Dire Wolf Alpha']
Decoding Process
Decoding Process

expr*  
?  
Next  Stop

Call
  func  args
  Name
    id
  "Aura"

Call
  func  args
  Name
    id
  "ChangeAttack"

Num
  n
  1

"MinionSelector"

Call
  func  Name
    id
  "Adjacent"
Decoding Process

Call
  func
    Name
      id
        "Aura"
  args
    Call
      func
        Name
          id
            "ChangeAttack"
      args
        Num
          n
            1
    Call
      func
        Name
          id
            "Adjacent"
Training
Training
Training
Training
Hearthstone Results
# Hearthstone Results

<table>
<thead>
<tr>
<th>Nearest</th>
<th>Ling et al. (2016)</th>
<th>ASN</th>
<th>ASN +SupAtt</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hearthstone Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Exact Match Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest</td>
<td>3.0</td>
</tr>
<tr>
<td>Ling et al. (2016)</td>
<td>6.1</td>
</tr>
<tr>
<td>ASN</td>
<td></td>
</tr>
<tr>
<td>ASN + SupAtt</td>
<td></td>
</tr>
</tbody>
</table>
Hearthstone Results

Exact Match Accuracy (%)

- Nearest: 3.0
- Ling et al. (2016): 6.1
- ASN: 18.2
- ASN + SupAtt: 18.2
Hearthstone Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Exact Match Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest</td>
<td>3.0</td>
</tr>
<tr>
<td>Ling et al. (2016)</td>
<td>6.1</td>
</tr>
<tr>
<td>ASN</td>
<td>18.2</td>
</tr>
<tr>
<td>ASN + SupAtt</td>
<td>22.7</td>
</tr>
</tbody>
</table>
Hearthstone Results

Token BLEU

- Nearest: 65.0
- Ling et al. (2016): 67.1
- ASN: 77.6
- ASN + SupAtt: 79.2
Hearthstone Results

Tree F1

- Nearest: 65.7
- ASN: 72.4
- ASN + SupAtt: 75.6
Semantic Parsing

show me the fare from ci0 to ci1

$$\lambda x. \exists y. \text{from}(y, \text{ci0}) \land \text{to}(y, \text{ci0}) \land \text{equals}(\text{fare}(y), x)$$
Semantic Parsing Results

Geo
- WKZ14: 90.4
- DL16: 87.1
- ASN: 85.7
- ASN+SupAtt: 87.1

ATIS
- WKZ14: 91.3
- DL16: 84.6
- ASN: 85.3
- ASN+SupAtt: 87.1

Jobs
- LJK13: 90.7
- DL16: 90.0
- ASN: 91.4
- ASN+SupAtt: 92.9

[Wang, Kwiatkowski, Zettlemoyer 2014] [Dong and Lapata 2016] [Liang, Jordan, Klein 2013]
Conclusion

Code generation and similar systems benefit from:

– Representing structure in output space
– Modular network architectures
Thanks!